Exception Handling
Part - II

- Jayendra Khatod

Functions for Trapping Exceptions

- **SQLCODE**
  Returns the numeric value for the error code

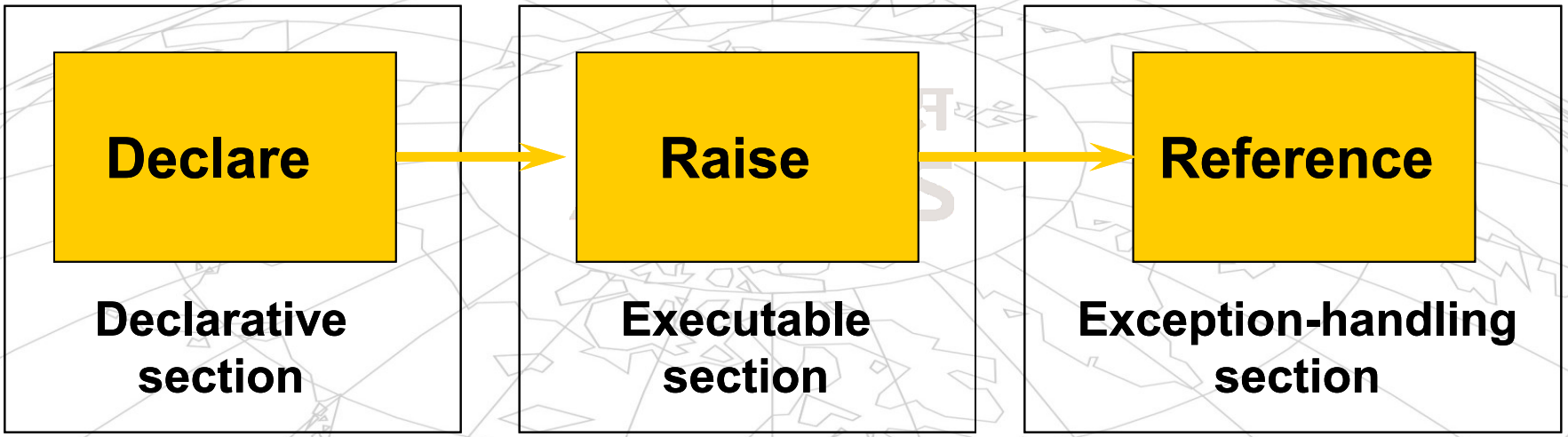
- **SQLERRM**
  Returns the message associated with the error number

Functions for Trapping Exceptions

- **Example**

```
DECLARE
   v_error_code       NUMBER;
   v_error_message    VARCHAR2(255);
BEGIN
...
EXCEPTION
...
   WHEN OTHERS THEN
      ROLLBACK;
      v_error_code := SQLCODE ;
      v_error_message := SQLERRM ;
      INSERT INTO errors
      VALUES(v_error_code, v_error_message);
END;
```

Trapping User-Defined Exceptions



- Name the exception
- Explicitly raise the exception by using the RAISE statement
- Handle the raised exception

User-Defined Exception

Example

```
DECLARE
   e_invalid_product  EXCEPTION;
BEGIN
   UPDATE      product
   SET         descrip = '&product_description'
   WHERE       prodid = &product_number;
   IF SQL%NOTFOUND THEN
     RAISE e_invalid_product;
   END IF;
   COMMIT;
EXCEPTION
   WHEN  e_invalid_product  THEN
     DBMS_OUTPUT.PUT_LINE('Invalid product number.');
END;
```

Propagating Exceptions

Sub-blocks can handle an exception or pass the exception to the enclosing block.

```
DECLARE
   . . .
   e_no_rows   exception;
   e_integrity exception;

   PRAGMA EXCEPTION_INIT
    (e_integrity, -2292);
BEGIN
   FOR c_record IN emp_cursor LOOP
    BEGIN
      SELECT ...
      UPDATE ...
      IF SQL%NOTFOUND THEN
        RAISE e_no_rows;
      END IF;
    EXCEPTION
      WHEN e_integrity THEN ...
      WHEN e_no_rows THEN ...
    END;

  END LOOP;
EXCEPTION
   WHEN NO_DATA_FOUND THEN . . .
   WHEN TOO_MANY_ROWS THEN . . .
END;
```

RAISE_APPLICATION_ERROR

- **Syntax**

```
raise_application_error (error_number,
                         message);
```

  - **A procedure that lets you issue user-defined error messages from stored subprograms**
  - **Called only from an executing stored subprogram**

RAISE_APPLICATION_ERROR

- Used in two different places:
  - Executable section
  - Exception section
- Returns error conditions to the user in a manner consistent with other Oracle Server errors

RAISE_APPLICATION_ERROR

- **Executable section:**

```
BEGIN
    ...
    DELETE FROM employees
    WHERE manager_id = v_mgr;

    IF SQL%NOTFOUND THEN
        RAISE_APPLICATION_ERROR(-20202,
        'This is not a valid manager');
    END IF;
    ...
```

RAISE_APPLICATION_ERROR

- **Exception section:**

```
...
EXCEPTION
     WHEN NO_DATA_FOUND THEN
     RAISE_APPLICATION_ERROR (-20201,
     'Manager is not a valid employee.');
END;
```

Summary

- Exception types:
  - Predefined Oracle Server error
  - Non-predefined Oracle Server error
  - User-defined error
- Exception trapping
- Exception handling:
  - Trap the exception within the PL/SQL block.
  - Propagate the exception.

Thank You !